

Choosing The Right Printer

COMPUTE!

\$2.95
June
1984
Issue 49
Vol. 6, No. 6
\$2.25 UK \$3.25 Canada
02193
ISSN 0194-347X ©

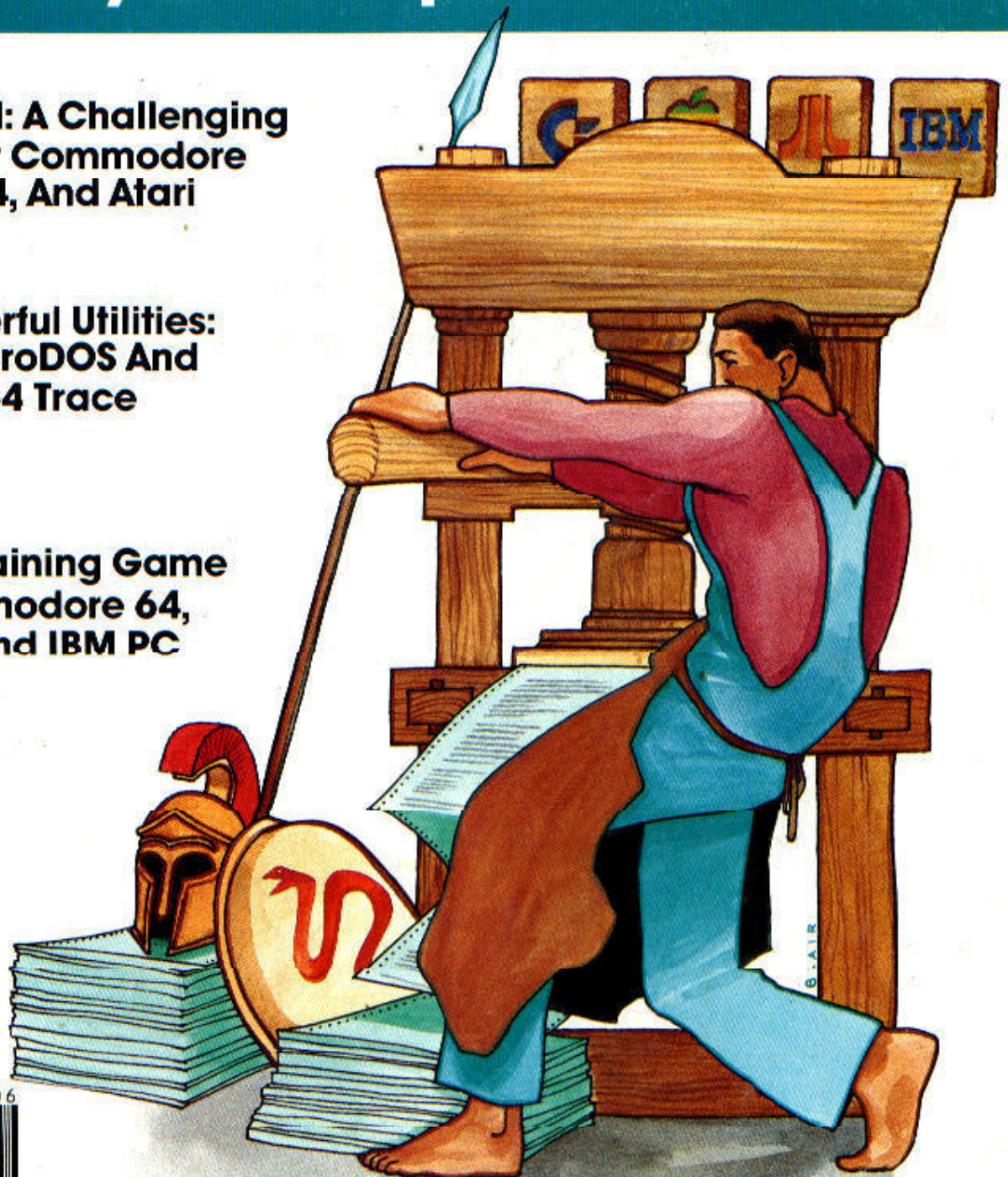
The Leading Magazine Of Home, Educational, And Recreational Computing

A Survey Of Inexpensive Printers

**Olympiad: A Challenging
Game For Commodore
VIC-20, 64, And Atari**

**Two Powerful Utilities:
Atari MacroDOS And
VIC And 64 Trace**

**Pests:
An Entertaining Game
For Commodore 64,
VIC-20, And IBM PC
And PCjr**



D

All the hits your computer is missing.



If you thought you'd never find fun games for your hardworking home computer, happy days are here. Because now ATARISOFT™ has all the great hits...Pac-Man! Donkey Kong! by Nintendo! Centipede! Defender! Joust! Jungle Hunt! Moon Patrol! Pole Position! Galaxian! Ms. Pac-Man! and Battlezone!™

And we've got them for all the hit computers...Apple, IBM, Commodore 64, Vic-20, Colecovision,* and TI 99/4A. We've got Pac-Man, Centipede and Defender for Intellivision too.

So dust off your joystick and ask your dealer for all the ATARISOFT hits. It's the software your hardware's been waiting for.

ATARISOFT.™

All the hits your computer is missing.

ATARISOFT products are manufactured by Atari, Inc. for use with various computers and video game consoles. ATARISOFT products are not made, licensed or approved by the manufacturer(s) of these computers and video game consoles. *Donkey Kong and Battlezone not available on Colecovision. 1. Trademarks of Baby Mfg. Co. Sublicensed to ATARI, Inc. by Namco America, Inc. 2. Trademarks and © Nintendo 1981-1983. 3. Trademarks and © Williams 1980-1982, manufactured under license from Williams Electronics. 4. Trademark and © of Taito America Corporation 1982. 5. Engineered and designed by Namco Ltd., manufactured under license by ATARI, Inc. Trademark and © Namco 1982. Atari®. © A Warner Communications Co. © 1984 ATARI, Inc. All rights reserved.

FEATURES

- 14 Choosing The Right Printer: The Easy Way To Hard Copy Selby Bateman
- 18 From Dot Matrix To Laser Print: The Changing Face Of Printers Selby Bateman
- 24 Bundling Printers With Computers: Did Coleco Answer A Need? Selby Bateman
- 26 The Inexpensive Printers Of 1984 Kathy Yakal
- 34 Avoiding Printer Problems J. Blake Lambert

EDUCATION AND RECREATION

- 36 Pests Kevin Woram
- 50 Olympiad Kevin Woram and Mike Buhidar, Jr.

REVIEWS

- 66 MailPro Elizabeth Deal
- 70 Promenade EPROM Programmer For VIC And 64 Sheldon Leemon
- 71 Stickybear Larry Ross
- 72 Two Games Of Strategy Dale F. Brown
- 74 Operation Whirlwind James V. Trunzo

COLUMNS AND DEPARTMENTS

- 6 The Editor's Notes Robert Lock
- 10 Readers' Feedback The Editors and Readers of COMPUTE!
- 78 On The Road With Fred D'Ignazio: The Morning After, Part 2 Fred D'Ignazio
- 82 Questions Beginners Ask Tom R. Halfhill
- 86 The Beginner's Page: A Wall Of Loops Richard Mansfield
- 90 Computers And Society David D. Thornburg
- 96 INSIGHT: Atari Bill Wilkinson
- 102 64 Explorer Larry Isaacs
- 106 Machine Language: A Program Critique, Part 3 Jim Butterfield
- 110 Programming The TI: TI Graphics C. Regina

THE JOURNAL

- 112 Commodore Information Handyman F. Joseph Walker
- 116 MacroDOS For Atari, Part 1 Jerry Allen
- 118 VIC And 64 TRACE Roger Harris
- 123 Apple Variable Save Jeff Brewster
- 126 Graphics 0 Text In Four Colors Ted Baldwin
- 129 Atari TAB Stephen Levy
- 131 Garbage Collection On Commodore Computers, Part 1 Jim Butterfield
- 134 Programming 64 Sound, Part 1 John Michael Lane
- 139 Apple Input And Menu Screens Dan Jordan

- 141 A Beginner's Guide To Typing In Programs
- 142 How To Type COMPUTE!'s Programs
- 144 CAPUTE! Modifications Or Corrections To Previous Articles
- 146 The Automatic Proofreader For VIC, 64, And Atari
- 149 News & Products
- 154 Product Mart
- 160 Advertisers Index

NOTE: See page 142 before typing in programs.

GUIDE TO ARTICLES AND PROGRAMS

	*
	*
	*
	*
	*
	*
	64N/PC/PCjr
	64N/AT
	V/64
	V/64
	AP
	64
	AT
	*
	*
	*
	*
	*
	*
	AT
	64
	64
	TI
	V/64/r
	AT
	V/64
	AP
	AT
	AT
	PN/64
	64
	AP

AP Apple AT Atari, F PET/ CBM, V VIC-20, C Radio Shack Color Computer, 64 Commodore 64, TS Timex/ Sinclair, TI Texas Instruments, PCr IBM PCjr, PC IBM PC, AD Coleco Adam. *All or several of the above.

**TOLL FREE Subscription Order Line
800-334-0868 (In NC 919-275-9809)**

COMPUTE! Publications, Inc. 
One of the ABC Publishing Companies

**One of the ABC Publishing Companies:
ABC Publishing, President, Robert G. Burton**
1330 Avenue of the Americas, New York, New York 10019

COMPUTE! The Journal for Progressive Computing (USPS: 537250) is published monthly by COMPUTE! Publications, Inc., P.O. Box 5406, Greensboro, NC 27403 USA. Phone: (919) 275-9809. Editorial Offices are located at 324 West Wendover Avenue, Greensboro, NC 27408. Domestic Subscriptions: 12 issues, \$24. Send subscription orders or change of address (P.O. form 3579) to **COMPUTE!** Magazine, P.O. Box 914, Farmingdale, NY 11737. Second class postage paid at Greensboro, NC 27403 and additional mailing offices. Entire contents copyright © 1984 by COMPUTE! Publications, Inc. All rights reserved, ISSN 0194-357X.

READERS' FEEDBACK

The Editors and Readers of COMPUTE!

How To Turn A Computer On

I have a question concerning peripheral equipment. When first turning on the computer equipment, I've heard that it is advisable to turn on the accessories first and the computer last. Is it okay to have all three units (computer, disk drive, and printer) plugged into a single power strip, and turn everything on at once merely by turning the power strip on?

Robert C. Leuten

No. Computers, and electronic equipment in general, often have circuits that protect against damaging surges of power when equipment is first turned on. By leaving all your equipment on and turning on the power strip, you defeat this circuitry. This could damage your equipment.

Also consider that the more devices on the power strip, the bigger the initial surge will be. So generally, it's a good idea to turn on each piece of equipment in the proper order, one at a time.

Another commonly asked question is, "In what order should I turn on the computer equipment?"

The Commodore 1541 disk drive owner's manual states that the computer should always be turned on last. Since the printing of that manual, Commodore has issued an update bulletin concerning the proper order for turning on the computer and its peripheral devices. Here are their recommendations:

1. Computer, disk drive, printer
2. Computer, disk drive, disk drive
3. Computer, disk drive, disk drive, printer

Variables In Atari Filenames

Is there any way you can assign a filename to A\$, and then open an Atari disk file named A\$?

James Beach

Sure. Let's say someone INPUTs the name into a string:

```
10 DIM T$(40),A$(20)
100 PRINT "Filename":INPUT T$
```

You can then create a disk filename:

```
110 A$="D:":A$(3)=T$
```

now we OPEN the file, for read access:

```
120 OPEN #1,4,0,A$
```

Disk Drive Door Dust Defense

I own a 1541 disk drive, and I would like to know if I should keep the disk drive door closed when it is not in use. I have read that if you keep the door closed, it will prevent dust from getting into the drive. On the other hand, I've also read that keeping the door closed also keeps the read/write head down, and the constant pressure will damage the head. Which would be better?

Jerrell F. Schivers

There is no compelling argument on either side of this debate. The pad that the read/write head rests on is soft, and shouldn't damage it with the door closed. On the other hand, dust can still find ways in with the door closed.

Tokenized Commands In TI Extended BASIC

Recently, I was working in Extended BASIC on my TI-99/4A and found that I could enter commands while in programming mode using the CTRL key. For instance, holding the CTRL key and pressing ; produces the PRINT command after the line is LISTed. (Note: This won't work in immediate mode or in console BASIC.)

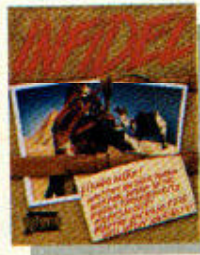
As it turns out, most keys in conjunction with the CTRL key produce a command. I've also discovered that only one such command can be entered per line in this fashion. Can you tell me the significance of all this?

Steve Hayner

Like most computers, TI represents its BASIC commands internally in a tokenized, or numerically-coded, abbreviated form. Apparently, certain keystrokes generate the same codes as some tokenized commands.

This technique is indeed limited to the Extended BASIC programming mode. Also, as you say, only one command can be entered per line with this method. These severe limitations, along with the absence of documentation in the TI-99/4A reference manuals, lead us to believe that the use of tokenized commands in this manner is allowed through a quirk in the system. They are probably not a design feature. Regardless, the method that you've described does offer a shortcut for entering commands in certain instances.





THE INCOMPLETE WORKS OF INFOCOM, INC.

Incomplete, yes. But it's not just because we're always bringing out new stories in the Infocom interactive fiction collection. Nor is it simply due to the fact that with all the writing and re-writing, honing and perfecting that we put into every one of our stories, our work is seemingly never done.

The real reason is: an Infocom work of fiction can never be complete until you become a part of it.

You see, as hard as we work at perfecting our stories, we always leave out one essential element—the main character. And that's where you enter in.

Once you've got Infocom's interactive fiction in your computer, you experience something akin to waking up inside a novel. You find yourself at the center of an exciting plot that continually challenges you with surprising twists, unique characters (many of whom possess extraordinarily developed personalities), and original, logical, often hilarious puzzles. Communication is carried on in the same way as it is in a novel—in prose. And interaction is easy—you type in full English sentences.

But there is this key difference between our tales and conventional novels: Infocom's interactive fiction is active, not passive. The course of events is shaped by the actions you choose to take. And you enjoy enormous freedom in your choice of actions—you have hundreds, even thousands of alternatives at every step. In fact, an Infocom interactive story is roughly the length of a short novel in content, but because you're actively engaged in the plot, your adventure can last for weeks and months.

In other words, only you can complete the works of Infocom, Inc. Because they're stories that grow out of your imagination.

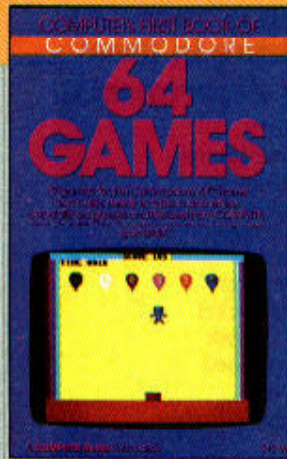
Find out what it's like to get inside a story. Get one from Infocom. Because with Infocom's interactive fiction, there's room for you on every disk.

INFOCOM™

Infocom, Inc., 55 Wheeler Street, Cambridge, MA 02138

For games: Apple II, Atari, Commodore 64, CD-ROM, IBM PC/compat, DEC Rainbow, DEC RT-11, IBM PC* and PCjr, KAYPRO II, MS-DOS 2.0*, NEC APC, NEC PC-8000, Osborne, Tandy 2000, TI Professional, TI 99/4A, TRS-80 Models I and III.

*Use the IBM PC version for your Compaq, and the MS-DOS 2.0 version for your Wang or Mindset.



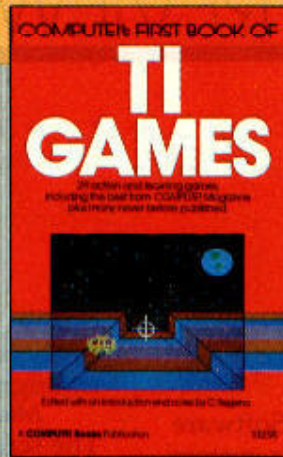
COMPUTE!'s First Book Of Commodore 64 Games

Packed full of games: "Snake Escape," "Oil Tycoon," "Laser Gunner," "Zuider Zee," and many more. Machine language games requiring fast hands and a good eye, as well as strategy games which will exercise your mind. Introductory chapters and annotated listings provide ideas and techniques for writing games. An excellent

production for 64 owners who want to begin writing games.

177 pages, paperback
Spiral bound for easy access to programs.

\$12.95
ISBN 0-942386-34-5



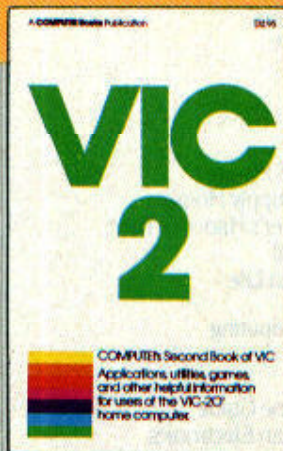
COMPUTE!'s First Book Of TI Games

Although this book is packed with ready-to-type-in games (29 in all), it is more than just a book of games. It is designed to teach game programming techniques. Introductory chapters explain the special features of the TI-99/4 and 99/4A, giving advice on coding techniques. Most games include an explanation of how the program works. Contains mazes,

chase games, old favorites, thinking games, creative challenges, and more.

211 pages, paperback
Spiral bound for easy access to programs.

\$12.95
ISBN 0-942386-17-5

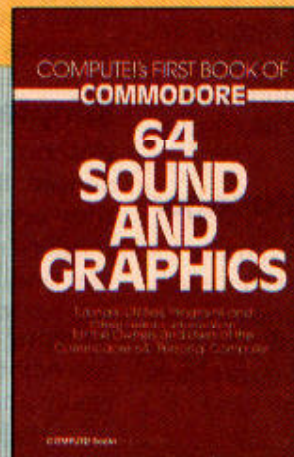


COMPUTE!'s Second Book Of VIC

This is just the book to follow the best-selling *First Book of VIC*: clear explanations of programming techniques, an extensive memory map, a mini word processor, a system for creating sound effects, a custom character maker, a machine language assembler, and "Gumball," an extraordinary all-machine-language game.

274 pages, paperback
Spiral bound for easy access to programs.

\$12.95
ISBN 0-942386-16-7



COMPUTE!'s First Book Of 64 Sound And Graphics

Clear explanations of the 64's sound and graphics capabilities. Includes many tutorials and example programs: "MusicMaster," a complete music synthesizer; "High-Resolution Sketchpad," an all-machine-language program for making computer art; and "Ultrafont Character Editor," one of the best character editors available. The appendices feature

useful reference charts and conversion tables.

275 pages, paperback
Spiral bound for easy access to programs.

\$12.95
ISBN 0-942386-21-3

COMPUTE! Publications, Inc. 

One of the ABC Publishing Companies

Post Office Box 5406, Greensboro, North Carolina 27403

THE BEGINNER'S PAGE

Richard Mansfield, Senior Editor

A Wall Of Loops

It takes most people a few weeks of part-time study to learn BASIC. Of course defined functions, multidimensional arrays, and other advanced techniques would not yet be understood, but after a short time, a novice programmer can accomplish a good deal with BASIC.

Nevertheless, during those first few weeks, most of us run into a wall—one of the fundamental BASIC commands is simply beyond understanding. Try as we might, some concept thoroughly resists our efforts to learn it. For me, the wall was the ON X GOTO 100,200,300 command. With furrowed brow, I came back to it again and again, trying to see how X controlled those line numbers following the GOTO.

Simple Loops

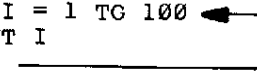
Others have said that their wall was *nested loops*. Let's take a look at these loops within loops.

Nested loops are one of the elements of computer power and a beginning programmer must be able to use them.

Here's a simple loop:

Program 1: Simple Looping

```
10 FOR I = 1 TO 100
20 PRINT I
30 NEXT I
```



The variable I is assigned a range of 1 to 100 in line 10. It is told that it will start out being a 1 and count up to 100 during the FOR-NEXT loop. And any commands between the FOR and the NEXT will be executed *each time* through this loop. In other words, line 20, which prints the current value of I, will be executed 100 times.

Anything else you want done 100 times can be squeezed in between lines 10 and 30 in this program. If you want your name printed 100 times, just put in a line 11 like this:

```
11 PRINT "MY NAME"
```

and it, too, will be printed. It's easy to see how this might come in handy when printing labels or addresses on a printer.

Now, to make the actions in Program 1 a bit

clearer, take a look at Program 2:

Program 2: Looping Without FOR-NEXT

```
10 I = 1
20 PRINT I
30 I = I + 1
40 IF I = 101 THEN END
50 GOTO 20
```

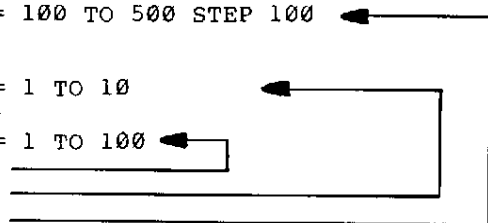
This does exactly the same thing as Program 1, but it's a bit clumsy. As you see, we *can* create a loop structure without using FOR-NEXT commands, but it takes up more room, takes longer to program, and runs more slowly. It's not generally the best way to set up loops, but it does help to visualize how a loop actually works.

Stuffed And Nested

Now we can try stuffing loops inside other loops. This is a technique which amplifies the power of loops. It's called *nesting* and the first FOR (coupled with the last NEXT) is called the *outer loop*:

Program 3: Nested Loops

```
10 FOR I = 100 TO 500 STEP 100
20 PRINT
30 PRINT I
40 FOR J = 1 TO 10
50 PRINT J;
60 FOR T = 1 TO 100
70 NEXT T
80 NEXT J
90 NEXT I
```



The outer loop in this program (the FOR in line 10 and the NEXT in line 90) causes the entire program to cycle five times, executing every command in lines 20-80 five times before stopping.

As an aside, the STEP command in line 10 is an interesting variation on the simple I=100 TO 500 command. Without the STEP, this program would execute 500 times. But STEP forces the I variable to add 100 to itself each time we hit the NEXT in line 90. So, instead of a series like 1,2,3,4,5,6,7 ... we get the series 100,200,300,400,500, a total of five cycles through the loop.

In any case, line 20 PRINTs a blank line, line 30 PRINTs the current value of the I variable, and then we come upon the first nested loop. The J

variable is given a range of 1 to 10, so everything between lines 40–80 will be performed ten times. But since this loop is nested inside the I loop (which creates five cycles of its own), the PRINTJ in line 50 will be executed 5 times 10. In other words, the value of J will be printed a total of 50 times in this program.

An even deeper loop, called the *inner loop*, appears between the FOR in line 60 and the NEXT in line 70. This loop is given a range of 1 to 100, but it isn't given anything to do. It just counts up to 100 and then we perform the NEXT J in line 80.

Do-Nothing Timers

That inner T loop does actually accomplish something, however. It uses up time. Such loops are often called *do-nothing loops* or *delay loops*. Their function is to slow down the computer. Sometimes this is very handy. Computers are fast. If you are having something PRINTed to the screen and it's sliding by too fast to read, insert a delay loop and give that loop whatever range suits your reading speed. Then, before allowing the program to proceed, the delay loop will count from the low up to the high number in its range.

Here is a second version of this same program, but, again, the FOR-NEXT commands are not used. If you are still unclear about how Program 3 functions, take a look at Program 4:

Program 4: Nested Loops Without FOR-NEXTs

```

10 I = 100
20 PRINT
30 PRINT I
40 J = 1
50 PRINT J;
60 T = 1
70 T = T + 1
80 IF T < > 100 THEN 70
90 J = J + 1
100 IF J < 11 THEN 50
110 I = I + 100
120 IF I = 600 THEN END
130 GOTO 20

```

Like Program 2, Program 4 is large, clumsy, and slow. For example, it takes five times as long to execute as Program 3, its counterpart. You'll probably never write nested loops like those found in Program 4, but you can take a look at it to see how nested loops are structured.

Program 4 also illustrates various true/false types of loop exits. Line 80 means that we keep on cycling through the loop if the variable T does not yet equal 100. We exit when T = 100. Line 100 continues to cycle as long as J is less than 11. In line 120, we exit the loop (and stop the entire program, via the END command) if I equals 600.

Rules And Customs

There are several programming rules and customs you should try to observe when working with loops. In general, a programmer cannot use the same variable name for different functions or the program might make serious errors. For example, if you are writing a program to figure out your budget and you say TAXES = 15000 (for federal tax) and then use the variable name TAXES again later in the program: TAXES = 400 (meaning state tax), you will have hopelessly confused the computer. You have to use different variable names, such as FED and STATE.

The same thing applies to loops. Each different loop must have its own name FOR I/NEXT I, FOR J/NEXT J, etc. To help keep this straight, most programmers use the variable I for their outer loop, then J, then K, and so on up the alphabet. The letters I, J, K, and L are *not used for normal variables*, just for loops. Similarly, the variable name T is reserved for timing loops, those delay loops we mentioned above.

Also, every FOR must have a matching NEXT to close its loop, and nested loops must not interweave. You cannot have a structure like this:

```


10 FOR I=1 TO 10
20 FOR J=1 TO 20
30 NEXT I
40 NEXT J

```

lines 30 and 40 are out of order. The inner loop, the J loop here, must be closed by its NEXT before the I loop can be closed.

VIC SOFTWARE 64

**More Games, Challenging Problems
and Programs Than You Can
Shake A Joystick At!**



FREE
1984
Catalog

FREE PROGRAMS

Write for Details.

ComputerMat • P.O. Box 1664C • Lake Havasu City, Arizona 86403

PROGRAMMING THE TI

C. Regena

TI Graphics

Drawing graphics is one of the things that really make our TI computers fun. Chapter 5 of the *Beginning BASIC* book that comes with your computer tells how you can get going with graphics. Using high-resolution graphics allows you to define your own characters and make detailed drawings on the screen. We can combine high-resolution graphics with text on the same screen, and we can use all 16 colors in high-resolution graphics if we wish.

There are several ways to define the graphics characters; this month we'll look at the most common ways. The CALL CHAR statement defines a certain character number with a certain pattern. If you use a number from 32 to 127, the regular symbol or letter will be redefined.

```
110 CALL CHAR(131,"3838107C10284282")
```

defines character number 131. Notice that the character definition pattern needs to be in quotes.

Using CALL CHAR

Another method is to define a string variable first, then use the CALL CHAR. This can save typing if you have several characters defined with the same shape:

```
150 A$="3838107C10284282"  
160 CALL CHAR(128,A$)  
170 CALL CHAR(136,A$)
```

If you have a lot of character definitions, DATA statements use less memory than many CALL CHAR statements. The disadvantage is that DATA statements are more difficult to type (and debug). This is an example:

```
200 FOR I=1 TO 10  
210 READ C,C$  
220 CALL CHAR(C,C$)  
230 NEXT I  
240 DATA 128,3838107C10284282,129,FFFF,130,FFFFFFFFFFFFFFFF,136,83E22618186447C1,141,204040808010102,142
```

```
250 DATA 20404080808C936,143,FFFF,144,01020408,145,0,151,FF
```

This loop defines ten characters, but instead of ten CALL CHAR statements, there are only six statements. This method is even more efficient when more graphics characters are defined. Within the loop, line 210 reads two values from the DATA statement (C and C\$). Line 220 uses these two values to define character number C with definition C\$.

If all your characters are in numerical order, you can use the character number as the loop counter. The DATA statements then contain only the definitions.

```
200 FOR C=97 TO 127  
210 READ C$  
220 CALL CHAR(C,C$)  
230 NEXT C  
240 DATA FFFF,,3838107C10284282,E0C8 (etc. for all the definitions)
```

Zeros Are Assumed

You can define a character with 16 numbers or letters (up to F). If you use fewer, the computer will automatically assume zeros for the rest of the definition. For example, FFFF really means FFFF000000000000. If you want to save memory and typing, arrange your graphics so the zeros are toward the bottom of the square defined. In other words, 0000FFFF00000000 and 000000000000FFFF and FFFF all look the same, but FFFF is the easiest to use. (The "bar" is positioned in different places in the graphics square.)

A character defined as null will be a blank square, or a square of the background color:

```
300 CALL CHAR(130,"")
```

In the DATA statement method, you can have commas with nothing between them:

```
310 DATA FFFF,,F0F
```

The middle definition is null. Both commas are

C3C8 00 BRK

Once again: Errors could be worked through in more detail. A BRK to the machine language monitor is not always explanatory.

```

;
; TRY AGAIN
;

```

C3C9 4C 00 C0 TA JMP CS

To do another file, we go back to the beginning of the program.

```

;
; FILE COPIED MESSAGE
;

```

```

C3CC 12 FCM .BYTE$12
C3CD 20 20 46 .ASC" FILE SUCCESSFULLY
               COPIED."
C3F2 0D 0D 12 .BYTE$0D,$0D,$12
C3F5 20 20 50 .ASC" PRESS RETURN TO COPY
               ANOTHER."
C419 0D 0D 12 .BYTE$0D,$0D,$12
C41C 20 20 50 .ASC" PRESS ANY OTHER KEY TO
               STOP."
C43B 0D 0D .BYTE$0D,$0D
C43D FCML = *-FCM

```

RAM Limits Are Set

Here are the limits of RAM for the program: They are arbitrarily set to allow space from \$4000 to \$7F00. I'm not sure why, but it's all right with me.

```

;
C43D 40 SP .BYTE$40 ;START GOREM
C43E 7F EP .BYTE$7F ;END GOREM
;

```

The following sequence is intended to initialize the disk. It does it in an unsatisfactory way: It opens the command channel again. (We have already opened the command channel as logical file 15.) The following code sends the BASIC equivalent of OPEN 1,8,15,"T":CLOSE 1. In a moment, I'll give a preferred approach.

```

;
; INIT DISK
;

```

```

C43F A9 01 ID LDA #INL
C441 A0 C4 LDY #>IN
C443 A2 5D LDX #<IN
C445 20 BD FF JSR SETNAM
C448 A9 01 LDA #1
C44A A2 08 LDX #8
C44C A0 0F LDY #15
C44E 20 BA FF JSR SETLFS
C451 20 C0 FF JSR OPEN
C454 20 CC FF JSR CLRCHN
C457 A9 01 LDA #1
C459 20 C3 FF JSR CLOSE
C45C 60 RTS

```

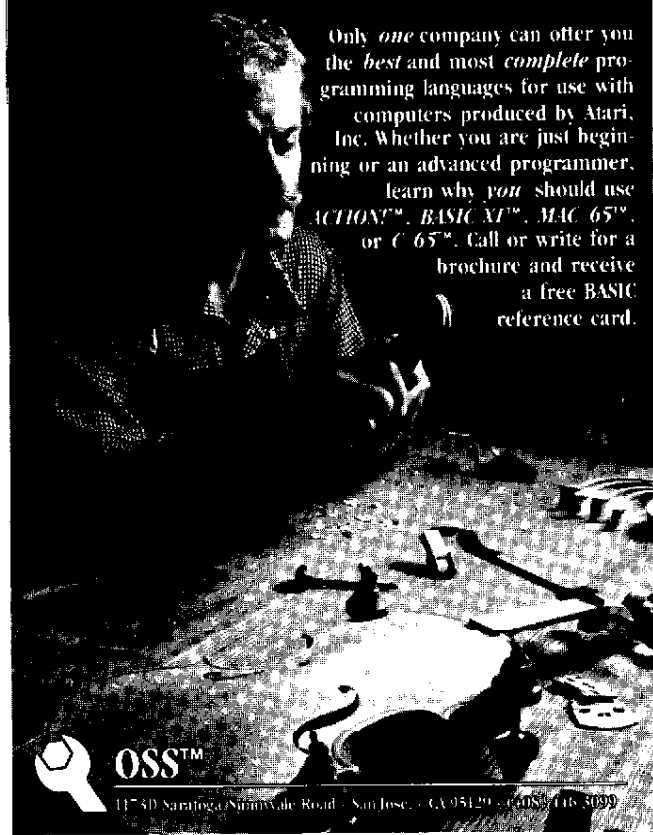
```

;
C45D 49 IN .ASC" T"
C45E INL = *-IN
;

```

What we should be doing is the BASIC equivalent of PRINT#15,"T", which is much easier:

PRECISION SOFTWARE TOOLS



Only *one* company can offer you the *best* and most *complete* programming languages for use with computers produced by Atari, Inc. Whether you are just beginning or an advanced programmer, learn why *you* should use *ACTION!*™, *BASIC XI*™, *MAC 65*™, or *C 65*™. Call or write for a brochure and receive a free BASIC reference card.



OSS™

1175D Saratoga Springs Road - San Jose, CA 95129 - (415) 416-8090

```

ID LDX #15 ;LF15, command
                channel
JSR $FFC9 ; .. connect to it
LDA #'T' ;Letter I
JSR $FFD2 ; .. send it
JSR $FFCC ;disconnect channel
RTS

```

Error Checking Needs Work

That's the program. It works reasonably well as given. The major improvements I would suggest are additional checking of the disk status (in the program given, the command channel was opened but never used); improved error message procedures; and a little rethinking of the RAM memory allocated.

The program has outstandingly clean documentation; it's a pleasure to read. In the same vein, the messages to the user are good and quite supportive. The coding approach is good, almost classical, in its methodical use of Kernal sub-routines. There's a lot to be learned from what's in the program, as well as from what's missing.

I'd like to thank Bud Rasmussen for allowing me to subject his program to analysis, warts and all. It can be embarrassing to have your mistakes—or your style—exposed to public view. I chose to pick through the program in detail because it was well-planned and well-written. Its faults are minor compared to its virtues.

vital. This particular DATA statement contains three definition strings.

Likely Errors

I mentioned that the data method of defining characters is more difficult to debug. If there is a problem, the most likely message is

BAD VALUE IN 220

You could also get the message

DATA ERROR IN 210

or

OUT OF DATA IN 210

Usually the typing in lines 210 and 220 is fine—the typing error is in the DATA statements. The DATA error messages occur if you don't have the commas placed correctly or if you're reading a string when it should be a number. The BAD VALUE message occurs because the program cannot define the character with what you have read in as data.

The easiest way to find the error is to RUN the program, then when it stops with the error message, print the variables involved. In this case PRINT C,C\$ and press ENTER to see what values we have for those variables. You should be able to see exactly what is wrong with your variables. C will tell you how far in the loop you got. Perhaps C\$ will have the letter O instead of the number zero, or maybe you've typed a period instead of a comma. In any case, you should be able to spot that error among your DATA statements so it can be corrected.

The CALL CHAR statement only defines the graphics character; you need to put the character on the screen using CALL HCHAR, CALL VCHAR, or PRINT. If a character is already on the screen and you use CALL CHAR to redefine it, all the characters on the screen with that character number will instantly change.

Changes On The Screen

Here's an example of changing character definitions while something is on the screen. Type this short program in, then RUN it.

```
100 PRINT "ABCDABCD"
110 FOR DELAY=1 TO 400
120 NEXT DELAY
130 CALL CHAR(65,"00666600422418")
140 FOR DELAY=1 TO 400
150 NEXT DELAY
160 END
```

The screen turns green when the program starts to run, and ABCDABCD is printed on the screen. After a delay loop, line 130 redefines character 65, which is the letter A. All the A's on the screen change. After another delay, the program ends. This technique might be useful to you in game situations when you want to change the graphics quickly.

I use a similar principle to PRINT graphics a

little more quickly than using CALL HCHAR or CALL VCHAR (as long as you don't have to worry about scrolling). Redefine as graphics the characters 96 through 126. Now, instead of using several CALL HCHAR statements to put the graphics on the screen, use PRINT with the lowercase letters. Suppose you have a snake defined in six graphics characters, 97 to 102. You can use PRINT "abcdef" to draw the snake on the screen.

Using Lowercase Letters

Release the ALPHA LOCK key to type the lowercase letters (which are actually small capital letters). Use FCTN and the key to type any symbol on the fronts of the keys. The reason you can use characters 96 through 126 so often in programs is that you may rarely need the symbols or lowercase letters in the text within a program.

To use characters from 129 to 159 in this PRINT method, look at the CONTROL KEY CODES list on your Reference Card (or in the Appendix of the *User's Reference Guide*). You can still PRINT graphics and in the quotes use the control key and the appropriate letter for the character number you want. You'll see either a blank or a funny graphics character as you're typing, but it will work fine in the program.


Every so often I read an article complaining that the TI does not have the capability to print graphics using built-in graphics characters or character strings. My rebuttal is that we *do* have the means to PRINT graphics, but we are not limited to graphics shown on the keys (such as on VIC-20, MC-10, or Timex graphics keys). We can define high-resolution graphics any way we wish, then PRINT the graphics using either lowercase letters, symbols, control characters, or CHR\$.

Changes For The TI-99/4

A special note to TI-99/4 (square-keyed console) owners: You cannot type in listings using lowercase letters, but a program typed on the TI-99/4A will work on the TI-99/4. If you don't have access to the 4A console, you can convert the PRINT statements by using the ASCII codes of the lowercase letters. 96 is ` (grave), then the lowercase letters start with 97 and go to 122. Instead of PRINT "abcdef", you can use

```
PRINT CHR$(97)&CHR$(98)&CHR$(99)&CHR$(100)
&CHR$(101)&CHR$(102)
```

You may use either the ampersand (&) or semicolons between the character numbers.

Our characters are grouped by eights into character sets which are used in defining colors. We use the CALL COLOR statement to define foreground and background colors for a particular character set—then all characters in that set will be the specified colors. If you need lots of colors on the screen, use different character sets. 

CAPUTE!

Modifications Or Corrections To Previous Articles

VIC Worm Of Bemer

The listing for the VIC version of this game from the April issue (p. 74) contains Commodore 64 color codes which are not available on the VIC. These cause no serious problems, but the [6] or [8] character should be omitted in lines 7715, 7730, and 10000.

Super Directory For 64 And IBM

Commodore 64 users have found that using "Super Directory" (Program 1, p. 173) from the April issue to load and run programs can cause problems if the program selected uses the BASIC function RND. An overflow error will be encountered because Super Directory alters a memory location used in calculating random numbers. Brian T. Bennett has discovered that the problem can be solved by changing line 1150 to:

```
1150 POKE 139,128:GOTO 5000
```

The IBM version (Program 4, p. 176) cannot be used to load and run programs from a disk with the write-protect notch covered. This is due to the way DOS handles the Write-Protect Error. Note also that the program as presented will work only with DOS 2.0 or 2.1.

TI Mozart Machine

Music aficionados may have detected a sour note in the tunes played by the TI version of this program from the January issue (Program 4, p. 168). The solution is to change the next to the last DATA element in line 480 from 287 to 587. Thanks to Kevin M. Norberg for this correction.

Atari Roader Improvements

Mike La Fave offers the following revision to this game from the March issue (Program 3, p. 70) to allow you to steer your racer with a joystick instead of the keyboard:

```
220 P=STICK(0):IF P=11 THEN N=N-1:GOTO 240
230 IF P=7 THEN N=N+1
```

Also, Keith Christleib suggests the following additions to include an engine sound as the car speeds down the track:

```
201 SOUND 3,135,2,9
315 SOUND 3,0,0,0
```

64Key Relocated

Reader Mike Levesque notes that the "64Key" program from the February issue (p. 160) uses the same area of memory as the DOS Wedge program supplied with the 1541 demo disk. To use these two valuable utilities together, he suggests changing the following lines:

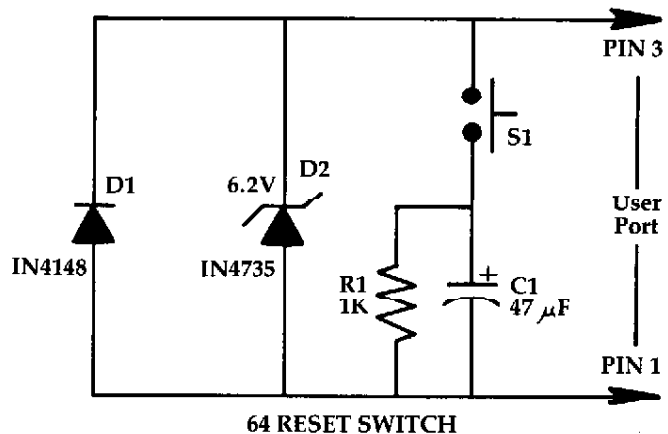
```
20 FOR I=51789 TO 51967
50 IF X<>23734 THEN PRINT "THERE IS AN
  ERROK IN YOUR DATA STATEMENTS":END
60 PRINT"SYS 51789 TO ACTIVATE":END
```

Next, change the DATA element 205 to 202 in the following lines: 100, 120, 130, 140, 150, 190, 220, 300, and 320. Finally, remove the ,0 from the end of line 430 and delete line 440. These changes relocate 64Key to the area immediately above the Wedge, allowing the two to coexist in harmony and still leaving locations 49152-51788 free for other uses.

64 Explorer RESET Switch

Columnist Larry Isaacs recommends a revision of RESET switch circuit for the 64 featured in his March column (p. 172). Larry based his design on the schematic diagram of the 64 included in the *Programmer's Reference Guide*. However, the actual circuitry in the 64 has since been slightly modified and, as a result, it is no longer safe to ground the RESET line directly. Although Larry has used his switch for several months without incident, it presents some risk of damaging the chips inside the computer, and you should consider this before attempting to use the switch on your computer.

As an alternative, Lester Iwamasa of Custom Concepts, who pointed out the danger of using the original circuit, has provided the following circuit which performs a RESET without the possibility of damage to the computer:



If you're not up to building this circuit yourself, you can obtain it for \$21.95, plus \$2.00 shipping, from:

Custom Concepts
30117 3rd Pl. SW
Federal Way, WA 98003

modore 64 and Atari 400 and 800 computers.

Gladstone Electronics, Inc.
1585 Kenmore Avenue
Buffalo, NY 14217
(716) 874-5510

TI-99/4A Cartridge Expander

Navarone Industries produces the Cartridge Expander, which plugs into the game port of the TI-99/4A and allows up to three cartridges to be plugged in at one time.

The expander also contains a built-in reset button and a select switch that lets you change from one cartridge to another without plugging and unplugging cartridges.

The Cartridge Expander is available for \$39.95.

Navarone Industries
510 Lawrence Expressway #800
Sunnyvale, CA 94086
(408) 866-8579

PCjr, Atari Audio Tutorials

Tutorials for new owners of PCjr and Atari 600XL and 800XL computers are available on audio cassette from FlipTrack Learning Systems.

How To Operate the IBM PCjr has two audio cassettes. The first cassette guides the user through start-up procedures; keyboard familiarization; simple BASIC programming; and the PCjr's color, sound, graphics, and mathematical capabilities, as well as cassette tape storage and use of a printer.

The second cassette includes information on how to manage disk storage and files with DOS. The lesson covers directory display, using tree-structured directories, checking disk storage space, and copying the formatting disks, as well as copying,

renaming, and erasing files, and batch processing.

How To Operate the Atari 600XL and 800XL Home Computer is a tutorial on one audio cassette and one data cassette. The package teaches start-up procedures, keyboard familiarization, and how to take advantage of the Atari's color, sound, graphics, and mathematical capabilities. Step-by-step BASIC programming is also taught.

The tutorials use the FlipTrack cassette format, which permits the user to branch into optional special interest areas with the flip of a cassette.

The PCjr tutorial sells for \$39.95, and the Atari tutorial is available for \$19.95. They operate on standard cassette players.

FlipTrack Learning Systems
999 Main
Suite 200
Glen Ellyn, IL 60137
(312) 790-1117

Four Educational Games For 64, Atari

Spinnaker Software has four new educational software titles, two for the Commodore 64, one for the Atari, and one for both computers.

Grandma's House, directed toward children four to eight, is a game for the 64 and the Atari that lets youngsters create and furnish their own playhouse. The program helps children learn to design and create, and is available on disk for \$34.95.

Ranch, ages five to ten, is available on cartridge for the 64. The program lets a player create and animate wild west scenes. Starting with a blank screen, the player populates it with a range of people, objects, and animals. You can copy, color, move, erase, or animate shapes. *Ranch* is

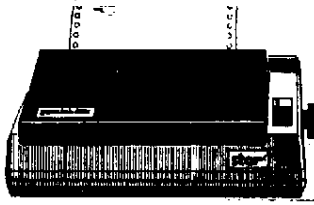
HUGE SAVINGS ON PRINTERS

FRICION (Single Sheet) & TRACTOR (Pin Feed)

GEMINI 10X
\$275.00

Star
ELEKTEK

DELTA 10
\$390.00



- 120 CPS
- Centronics Parallel interfaced



- 160 CPS
- 8K Buffer (Exp. to 16K)
- Both Parallel/serial interfaces are standard

GEMINI 15X

Same great features as above
15" wide carriage **\$400.00**

Delta 15

Same great features as above
15" wide carriage **CALL**

CABLES/ACCESSORIES

PA10A 10 ft. 36/36 pin standard parallel	32.00
IB-P10 10 ft. 36/25 pin parallel for IBM	32.00
PA6T 6 ft. 36/16 pin parallel for TI-99/4A	25.00
RS10A 10 ft. 25 pin standard RS-232 (full loaded)	25.00
APPLE DUMPLING GX	65.00
GRAPPLER PLUS	105.00
Buffered (16K) GRAPPLER PLUS	165.00
Star Universal Communications Interface	55.00
Gem-05 Serial Interface	100.00
Gem-01 Replacement Ribbon	6/15.00; 12/24.00
Elek-Tek Dust Covers for Gemini 10X/15X	5.00

Corp. Accts. Invited. Min. Ord. \$15.00. VISA/MC or phone. Mail Cashier's Check, Money Order, Personal Check (2 wks. prior). Add \$4.00 per item (A.K., H.I., P.R.). Canada add \$10.00 first item. \$1.00 ea. add. shipping & handling. Shipments to IL address add 7% tax. Prices subject to change. Write for free catalog. Return policy: Replacement only for defective on arrival. Freeowner, MFR. Warranty applies. **ALL ELEKTEK MERCHANDISE IS BRAND NEW, FIRST QUALITY AND COMPLETE.**

ELEK-TEK, inc.

6537 N. LINCOLN AVE., CHICAGO IL 60645
(800) 621-1269 (312) 677-7660